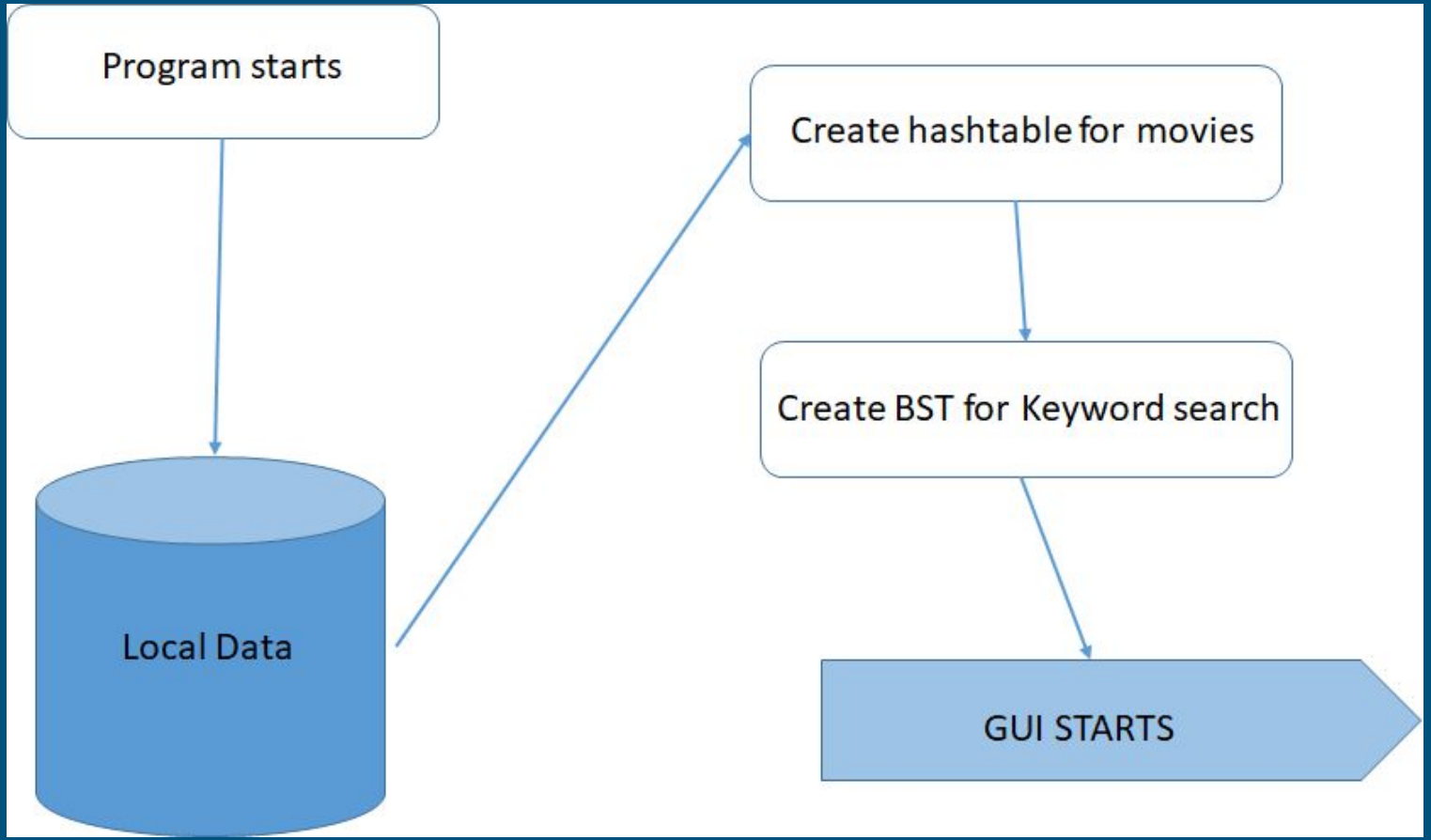
Not Netflix movie database

Henry, John, Ben

Overview

- Database to store information about movies
- Movie objects
  - Title
  - Year
  - Genre
  - Rating
  - ID
  - Runtime
- User can search with different keywords

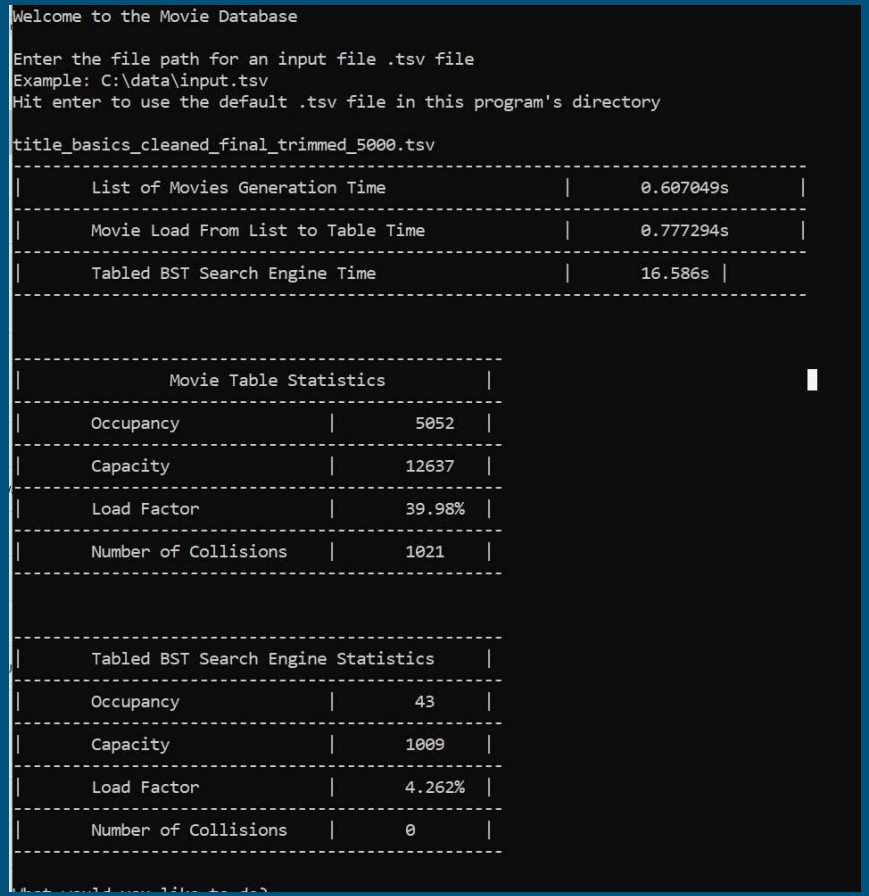


Program workflow

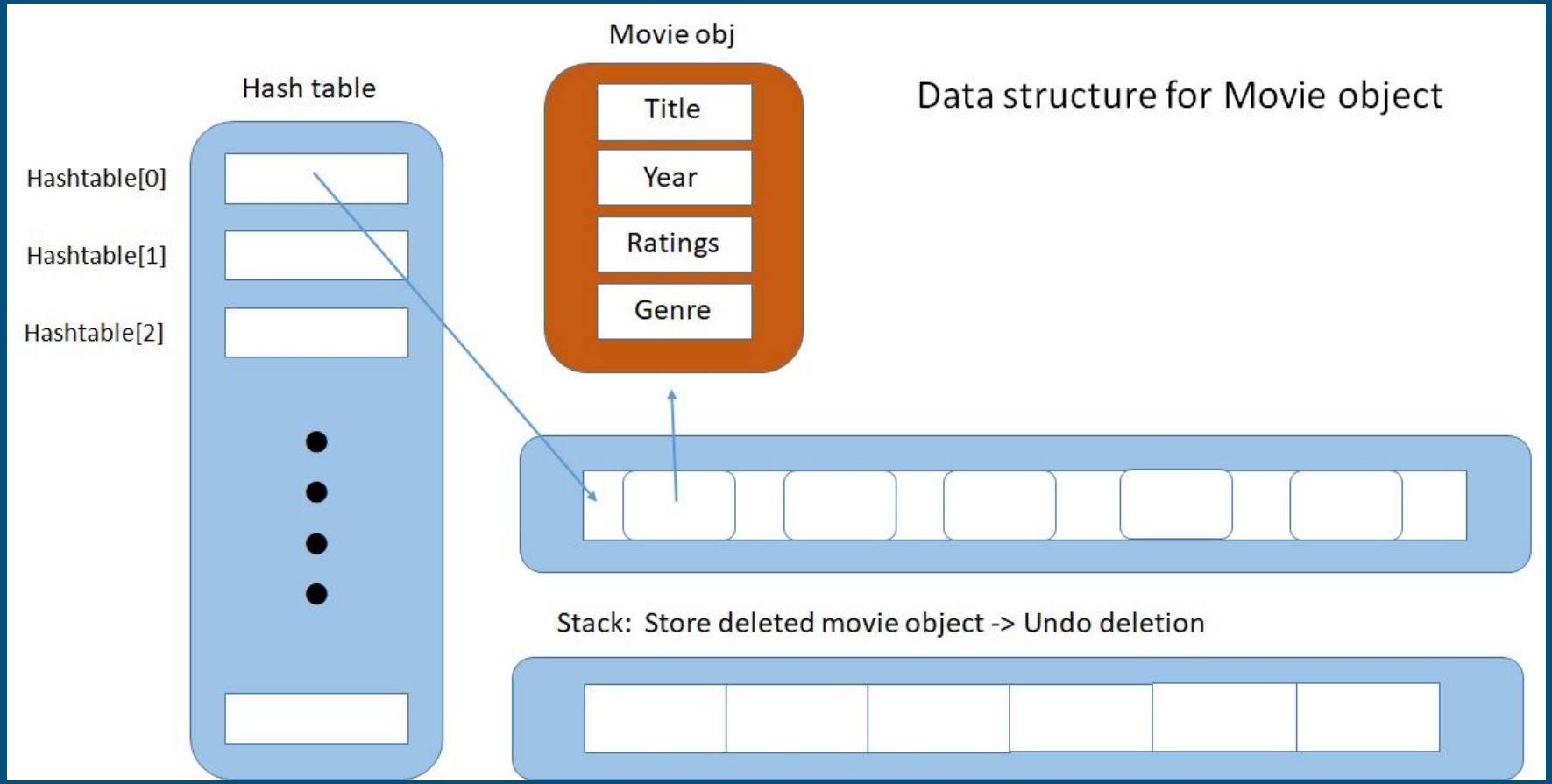
Menu operations

- Search for a movie
  - View related movies
- Add a new movie
- Remove a movie
- Edit a movie
  - Pick from potential matches
- Undo a deletion
- Exit program

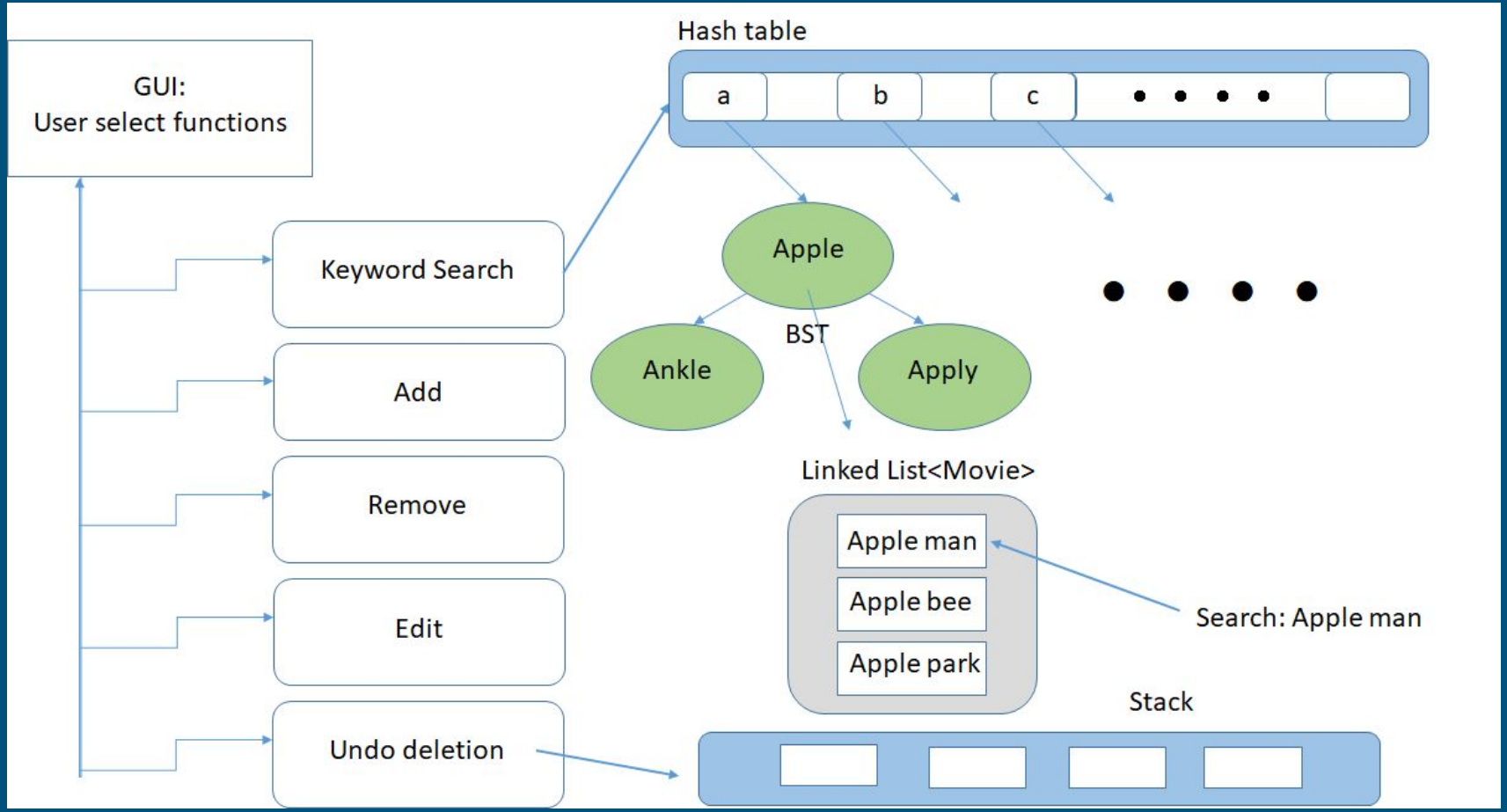
Data Structures



Performance



Structure diagram



Structure chart

Database Wrapper

```
HashTable<BinarySearchTree<string, Movie>*>*__searchEngineBST

HashTable<Movie>*>*__movieDB

Stack<Movie>*>*__deletedMovies
```

Binary Search Tree

```
template <typename K, typename V>
class BinaryNode {
private:
    K               __key;
    List<V>*        __values;
    BinaryNode*     __leftChild;
    BinaryNode*     __rightChild;
```

Binary Search Tree

- Adding a new tree

```
__searchEngineBST->add(std::string(1,
    keywords->getEntry(j)[0]),
    new BinarySearchTree<std::string,Movie*>(keywords->getEntry(j)));
```

- Updating the engine

```
(*__searchEngineBST)[firstCharOfKeyword]->addValue(keyword, edittedMovie);
```

Hash Table

- Open addressing
  - Quadratic probing for collision resolution

```
h = (h << 4) + *sp;
if (g = h & 0xF0000000) {
    h ^= g >> 24;
    h &= ~g;
}
```

```
hashID = (hashID + size_t(0.5 * i) + size_t(0.25 * i * i)) % __capacity;
```

Hash Table

- Updating an attribute

```
std::string processedKey = processSearchEntry(key);
Movie newMovie = __updateSearchEngineBST(
    newYearReleased, (*__movieDB)[processedKey], 2);
(*__movieDB)[processedKey] = new Movie(newMovie);
```

Linked List

- Deletion handler

```
std::string keyFound =
    __searchEngineBST->get(firstCharOfKeyword)->getKey(keyword);
List<Movie*>* keywordMovies =
    __searchEngineBST->get(firstCharOfKeyword)->getValues(keyFound);
keywordMovies->removeByValue(movieToDelete);
```

Demo

Q&A